Jack Girard
ECE 637
Lab 8: Image Halftoning

1. Introduction

No Deliverables

2. Image Fidelity Metrics

No Deliverables

3. Thresholding and Random Noise Binarization

1. The original image and the result of thresholding

Figure 1: Original house.tif Image

Figure 2: Thresholded house.tif Image

2. The computed RMSE and fidelity values

RMSE	Fidelity
87.3933	77.3371

3. Code for *fidelity* function - *See Next Page*

```python
def fidelity(f, b):
    # calculate dimensions of input images
    dim1, dim2 = np.shape(f)

    # gamma correct input images
    gamma = 2.2
    fl = 255*(f/255)**gamma
    bl = 255*(b/255)**gamma

    # define 7x7 Gaussian filter
    sigsq = 2
    size = 7
    h = np.zeros((size, size))
    for i in range(-3, 4):
        for j in range(-3, 4):
            h[i+3,j+3] = np.exp(-(i**2 + j**2)/(2*sigsq))
    sumh = np.sum(h)
    C = 1/sumh
    h = C*h

    # pad gamma un-corrected matrices and initialize new matrices
    fl = np.pad(fl,((3,3),(3,3)))
    bl = np.pad(bl,((3,3),(3,3)))
    flpf = np.zeros((dim1,dim2))
    blpf = np.zeros((dim1,dim2))

    # apply 7x7 Gaussian filter to each index
    for i in range(3, dim1+3):
        for j in range(3, dim2+3):
            flpf[i-3,j-3] = np.sum(np.multiply(fl[i-3:i+4, j-3:j+4], h))
            blpf[i-3,j-3] = np.sum(np.multiply(bl[i-3:i+4, j-3:j+4], h))

    # re-gamma correct
    ftil = 255*((flpf/255)**(1/3))
    btil = 255*((blpf/255)**(1/3))

    # calculate fidelity
    count = 0
    for i in range(dim1):
        for j in range(dim2):
            count += (ftil[i,j] - btil[i,j])**2
    fid = np.sqrt((1/(dim1*dim2))*count)

    return fid
```

4. Ordered Dithering

1. The three Bayer index matrices of sizes 2 × 2, 4 × 4, and 8 × 8

$$2 \times 2\ Bayer\text{:}\ \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

$$4 \times 4\ Bayer\text{:}\ \begin{bmatrix} 5 & 9 & 6 & 10 \\ 13 & 1 & 14 & 2 \\ 7 & 11 & 4 & 8 \\ 15 & 3 & 12 & 0 \end{bmatrix}$$

$$8 \times 8\ Bayer\text{:}\ \begin{bmatrix} 21 & 37 & 25 & 41 & 22 & 38 & 26 & 42 \\ 53 & 5 & 57 & 9 & 54 & 6 & 58 & 10 \\ 29 & 45 & 17 & 33 & 30 & 46 & 18 & 34 \\ 61 & 13 & 49 & 1 & 62 & 14 & 50 & 2 \\ 23 & 39 & 27 & 43 & 20 & 36 & 24 & 40 \\ 55 & 7 & 59 & 11 & 52 & 4 & 56 & 8 \\ 31 & 47 & 19 & 35 & 28 & 44 & 16 & 32 \\ 63 & 15 & 51 & 3 & 60 & 12 & 48 & 0 \end{bmatrix}$$

2. The three halftoned images produced by the three dither patterns

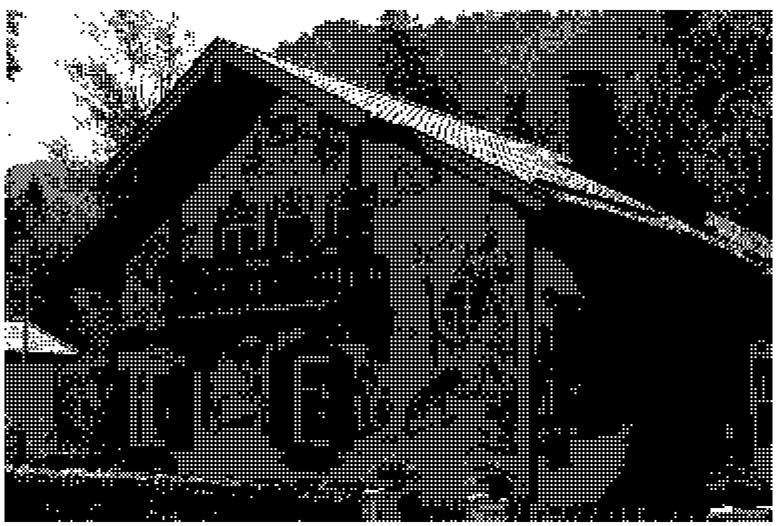

Figure 3: 2×2 Bayer Ordered Dither Halftone of house.tif

Figure 4: 4×4 Bayer Ordered Dither Halftone of house.tif

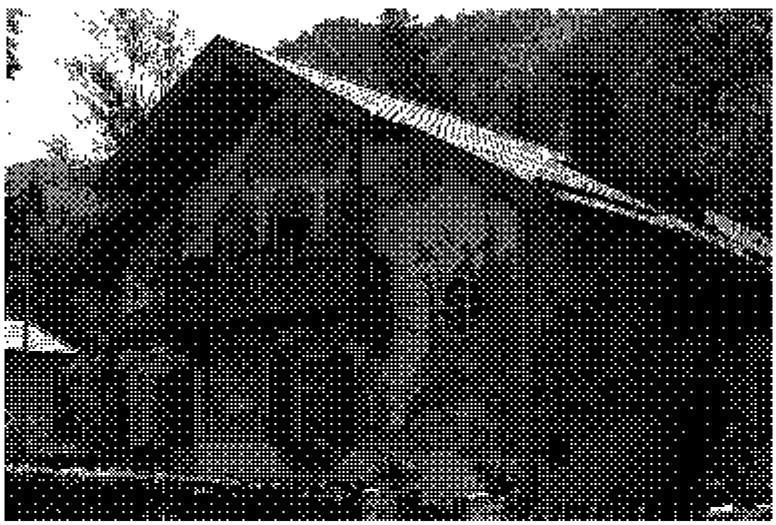

Figure 5: 8×8 Bayer Ordered Dither Halftone of house.tif

3. The RMSE and fidelity of each of the three halftoned images

Bayer Threshold Matrix Used	**RMSE**	**Fidelity**
2 × 2	97.6690	50.0569
4 × 4	101.0069	16.5583
8 × 8	100.9145	14.6918

5. Error Diffusion

1. Error Diffusion Python Code

* See Next Page ↓ *

2. Error Diffusion Result

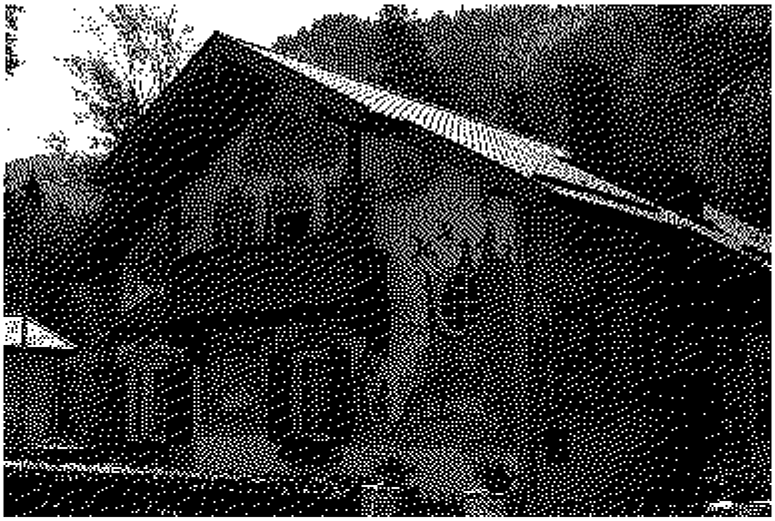

Figure 6: Error Diffusion Halftone of house.tif

3. RMSE and fidelity of the error diffusion result

RMSE	Fidelity
98.8471	13.4273

4. Tabulating the RMSE and fidelity results from the Lab

	Threshold	2×2 Bayer Dither	4×4 Bayer Dither	8×8 Bayer Dither	Error Diffusion
RMSE	87.3933	97.6690	101.0069	100.9145	98.8471
Fidelity	77.3371	50.0569	16.5583	14.6918	13.4273

Comments & Observations:

Between the different halftoning methods, the RMSE does not change significantly. In fact, as more sophisticated methods are used the RMSE increases slightly. The fidelity varies significantly between the different halftoning methods. The thresholded image has the largest fidelity, but its quality is not great and many features from the original image are not preserved. As the Bayer dither window size is increased or if error diffusion is used, the fidelity decreases. However, these results contain more distinguishable features and are more representative of the original grayscale image when viewed from a slight distance. It appears that RMSE is not an effective metric to measure perceived image quality using the human eye, and lower fidelity corresponds to better visual representation of the original grayscale image.

In [5]:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from PIL import Image
import import_ipynb
import io
import fidelity

im = Image.open('house.tif')
f = np.array(im)
```

In [6]:

```
dim1, dim2 = np.shape(f)
gamma = 2.2
fl = 255*((f/255)**gamma)
```

In [7]:

```
T = 127
out = np.zeros((dim1,dim2))
out = np.pad(out,((1,1),(1,1)))
fl = np.pad(fl,((1,1),(1,1)))

for i in range(1,dim1+1):
    for j in range(1,dim2+1):
        # apply quantization
        if fl[i,j] > T:
            out[i,j] = 255
        else:
            out[i,j] = 0
        # compute quantization error
        pixelError = fl[i,j] - out[i,j]
        # diffusing error to other indices
        fl[i+1,j-1] += pixelError*(3/16)
        fl[i+1,j] += pixelError*(5/16)
        fl[i+1,j+1] += pixelError*(1/16)
        fl[i,j+1] += pixelError*(7/16)
# remove padding from output matrix
out = out[1:-1,1:-1]

# display image
# plt.figure()
# plt.title("Error Diffusion Halftone of house.tif")
# plt.imshow(out,cmap=plt.cm.gray,interpolation='none')

# save image as .tif
img_out = Image.fromarray(out.astype(np.uint8))
img_out.save("ErrorDiffusion.tif")
```

In [8]:

```
fid = fidelity.fidelity(f, out)

# compute RMSE
count = 0
for i in range(dim1):
    for j in range(dim2):
        count += (f[i,j] - out[i,j])**2
RMSE = np.sqrt((1/(dim1*dim2))*count)
```

```python
# display RMSE and fidelity
print("RMSE = ", RMSE)
print("fidelity = ", fid)
```

```
RMSE =  98.84711671109255
fidelity =  13.427253039026654
```